From:	Bill Gates
Sent:	Monday, October 07, 1996 8:06 PM
To:	Steven Sinofsky; Eric Michelman
Cc:	Tandy Trower; Nathan Myhrvold; Pete Higgins; Butler Lampson; Jon DeVaan; Chris Peters; Richard Fade; Joe Belfiore; Aaron Contorer; Brian MacDonald (Exchange); Jeff Raikes; Andrew Kwatinetz
Subject:	Thoughts on simplifying user interface

userinterface.doc

Thoughts on the Office User Interface

Its very easy to write a memo saying that our user interfaces should be easier. Its very hard to find a way to do so. This memo doesn't have the answers but it talks about some of the directions that might be fruitful. It lays out some challenges. Please feel free to forward this to Microsoft people who think about user interface.

I think there is a lot more merit to the current Office interface than most of us are willing to give credit. People can ridicule me when I say I like the Office interface but I think its an amazing piece of work. It not a typical Microsoft thing to praise what we are already doing well but it becomes worthwhile we are considering radical change. The great parts of the interface need to be preserved as we move forward. Microsoft invented tool bars and wizards and many other things to handle real scenarios. However as many people have noted we have to be even more clever in order for users to notice and use additional capabilities. One of our weaknesses is that we haven't put much energy into getting rid of user interface elements from the past. Our innovation has been largely additive.

The user assistant is the most exciting thing we have done recently.. Since the user assistant is fairly new there are a lot of times when users type something in and don't get a good response. However this is not because the concept is wrong. We need to have a set of users who record things they ask the assistant to do when they don't like the response. Its important to persevere in getting full value out of the assistant and not go off in a new direction leaving it half done. I think we should have at least tens of thousands of users who can generate an email message to us whenever they put something in the assistant and don't get back something helpful. We simply need a hook in the assistant for this and an "add-on" that you get on the Internet that enables the function when the person is on-line. This is live feedback at its finest.

Here are some areas where we should consider improvements.

Unification of applications

We have done a better job than any competitor of sharing between the different applications. However there is still a huge boundary to cross when you move between the applications. Particularly if you want to do something fairly simple like a table and you are already in Word its very tempting to just stay in Word and accept its limitations versus Excel. If you are creating anything larger than a page we have NEVER solved the tricky issue of allowing the compound object to print reasonably. Perhaps any solution would be too complex.

Lets say I am in a meeting and I want to take notes. I am making a list of things with various dates attached to them. Should I work in Excel, Word or Outlook? I want some of the richness of each of these. This notetaking scenario is a very common one that Jeff Raikes reminded me recently how poorly we are optimized for.

I would prefer to have a modal Word that knew when I was doing a presentation and when I was doing a letter than to have 2 different .EXEs. The advantage is that I know everything that is common will be shared - commands and code. I know that my styles will work in both cases. I know my glossaries will be available in both cases.

I am not saying it is easy to extend Word to do most everything Powerpoint does but I know our users would love it if we were able to achieve it. I don't think its totally out of line to study this merger as a possibility for Office 9X. The Escher work done for Office 97 puts us in a much better position to do this.

Looking further out I think FrontPage, Word, Publisher and Powerpoint will be built off of a common code base. We may not pull them all together at once but I think competition will force us to do so over time. My personal view has been that we should create a product with Frontpage, Powerpoint and Works-WP richness on top of Trident as part of our Webworks efforts.

Its hard to have a discussion about unification of applications because organizational, timing, and design issues all get mixed together and it very confusing. However I think we have to find a way to discuss unification.

Unification of concepts between applications and the system

There are many things inside our applications which similar to each other but different. Multi-user support is quite different. Outlining is quite different. Styles are somewhat different. The commands for creating a named area and managing that set of names is quite different (see Unification of tables of information discussed below).

There is a huge opportunity to design across the Windows/Office boundary. The commands for taking a "file" and moving it somewhere include save, post, publish/put and send because the platform has many storage systems. Even the clipboard is a special purpose "storage". Office could promote a unification of these storage systems that would start by hiding the differences from Office users.

User preferences are an area of opportunity. They should be included in the state which is easily stored on a server so that a user get the state whereever they log in. This requires making sure these preferences can adjust to hardware differences. Office needs to coordinate with the manageable PC effort in Systems.

Although the file system improved dramatically between Windows 3.1 and Windows 95 it is still the most opaque and confusing thing that users have to deal with. The Office group should not view this as simply a fact of life. They are in a position to help us move forward by suggesting how we make the file system appear. Clearly the ONLY parts of an application installation a user should see are those parts we want them to manipulate. In some places we use folders as another way to have a named set of items that they can add and delete from. It should be very easy to find the documents you create. We should provide just a few rigid approaches to document storage starting with the user who has so few they want them just on their desktop, followed by someone who wants them all in a folder, followed by someone who wants them by project (which is the general case).

We have a chance during calendar 1997 to completely shape how the IE4/shell integration is viewed. We can add rich behavior to the folders used for Office documents. This is controversial. With all of the limitations of the file system some people say that hiding it more and more is the best approach. I personally think a lot can be done with IE4/Shell. However it will require unifying what you see in the open dialog box with what you see when you run the shell. This means we have to invoke IE4 code to display directories when you browse them during use of open and other file system commands. Some good thinking has to be done on this. The sooner the Office group gets out in front of this the more our IE4/Shell message will provide real benefits. We risk it being a dud and a source of tension between IPTD and Office if we don't come up with some approach.

Unfortunately we have already merged everything that is easy to merge so additional concept merging is hard. I am very interested in other peoples lists of things that should be unified particularly if they think it is easy.

Unification of mechanisms

Xerox and Metaphor designed their productivity software using common verbs, property sheets and a form of templates very heavily. They did not have hardly any top level menu commands. Although they weren't as feature rich as our applications it was quite impressive how far they got with these tools. The verbs were OPEN, MOVE, COPY, and PROPERTIES. One great technique is recursively treating a data structure as a native document. Lets say you have a set of names in a word processor - instead of presenting a lot of commands to delete names, change names, copy names etc.. you build a document and have fairly intuitive mappings between editing this document and how the name table is changed. I admit this approach has it dangers - it can be over used but I think style sheets, auto correct table, glossaries, and bookmarks could all be handled this way reducing the number of commands and increasing the power of the application. All of our applications have tables like the ones described above and there is far too little unification of searching, merging, moving or anything else this part of the applications state. The power increases because all of the rich navigation that was built for normal documents is now available to help manage these other structures. The code doesn't get much a lot simpler - just the user interface. This is a very concrete suggestion so if it seems vague please force me to be more specific.

Another great idea from the Xerox days is their concept of "templates". This is SOMEWHAT DIFFERENT from our concept of template. Instead of having a command for making a rectangle they made sure that it was very easy to have another document open and to copy from the document. Whenever a new object was desired for the word processor they would simply add it to their master template. Users could easily customize which things were on which templates. Everything that you find today on our Insert menu would have been handled this way. To get a picture you would copy in their "generic picture" and then right click on it to get the dialog box to browse and find a picture. Of course to be totally rich you make it possible to take any object in a document and put it onto a menu/toolbar so if I like purple trapezoids I simply move that up onto a toolbar/menu. The user interface for calling up another document and copying from it would have to be improved in order to rely on this one very powerful mechanism so heavily.

MIT has a 3D sketching system called Sketch where they use the mouse to specify almost all of the commands. They have no menus or tool bars or dialogs of any kind. Dennis Adler in Research has a 15 minute video of this which I encourage everyone to look at. Although the cleverness here doesn't map directly onto our problems it does show how using a few very clever commands (or gestures) you can do something very powerful. Typical 3D drawing programs have dozens of commands where they have none.

Microsoft Office has a lot of mechanisms - menus, tool bars, status bar, dialog boxes (with tabs which are a symptom of being overly large - an easy criticism to make but harder to solve) and right click menus. We took a major step towards unifying menus and tool bars in Office 97 by putting the icon for the short tool bar form in with the text in the longer tool bar and menu form. We took a step towards unifying dialogs by allowing drop downs in tool bars/menus but we stopped short of unifying dialog boxes. Users will appreciate this unification a great deal. Lotus has done some great work unifying dialog boxes and tools bars. A key element of this is allowing dialog boxes to stay active and making it easy to move up and down the hierarchy of objects which the dialog box shows the state of. Another advance has to be dealing with dialog properties that are driven by styles rather than set directly - this is too complex today. The status bar should also be unified.

For many years Tandy Trower and I wondered if all top level menus could be replaced by right click menus. I love right click menus because they have context and they allow new objects with rich behavior to show up and not complicate things for people who don't use them. There are arguments against this. In order to "see" all the commands you have to go to a document we prepare with all the samples and right click on them. We need to improve right click so that it works when you have a selection that includes lots of different types of objects. If I want to color a set of things blue I should be able to select them and right click and see the common commands. This requires synthesizing the right click menu at run time. There is a real question how much benefit it is to take things off of the top level menus. The extreme of getting rid of the top level menus requires some cleverness to add a few objects to right click on. For example Tandy has been a strong advocate that the title bar when right clicked in the description area should give all of the document level commands which reside on the FILE menu today.

I believe the user assistant is one of our best mechanisms. I think people are too cynical about our users willingness to take some time to learn. I admit they don't want to do it all right when they install the application. However I think if the assistant mentioned say 10 topics that at some point they might want to "learn" later on and made it easy for them to go back and find that list we would be surprised how many users would get around to it. Each topic would have to be less than 10 minutes and use the audio richness web tutorials can now deliver. These lessons would be suggested when the user is having trouble with something. We also want to encourage users to look at rich documents created with Office and be able to say "Show me how this was done". Right now I get sample document but they are buried and when I look at them there is no easy mapping back to the concepts that I need to learn. I think some users will respond to a set of tests of proficiency where I can say to myself that I know Excel 97 very well and check it out. This would respond to notion that users have to be retrained in an expensive fashion. This should scale into the full blown certification program (Microsoft On-Line Institute) in a form where you don't have to go to a testing center - you do on line.

Recording user behavior and doing deep pattern matching

I believe we have not taken as much advantage of watching user behavior in using our applications as we can. Its great that we have done some statistical work and some observation work but we shouldn't think that we have come even close to taking full advantage of these approaches. Even in documenting editing where we have done our best work there are still common formatting, text movement and navigation jobs that are fairly hard.

Ultimately our applications will record the entire history of user behavior and find patterns the same way a human looking over your shoulder would. We did some prototypes using the bayesian inference engine that research has. Unfortunately our model of user patterns and our ability to "match" against them are very weak. I think it will take a great engineer thinking a lot about user actions and how to match against it before we move to the next level here. The ideal would be for someone (like program management) to write up about 50 things that someone watching you use a productivity applications would suggest that go beyond current Intellisense capability and challenge engineering to see if there is a way to record and match so those suggestions can be made by the computer. Progress in this area requires research and Office to work together.

New input techniques

The "Holy Grail" of user interface is the computer that sees you and listens to you just like a real assistant. Within a decade a lot of this capability will be standard inside Windows. The language parsing in the assistant is an important step in this direction. By improving our ability to handle typed questions in the assistant we prepare ourselves for a speech driven system. Although I don't expect speech input to be common for productivity tool users within the next 3 years we need to track the progress in the research group and make sure our user interface direction will fit into their work. I think within 6 years these techniques will be a critical part of the interface which means it not too early to include this in our strategic thinking. We will be increasing our research budget aggressively focusing on exactly these areas because we think operating systems will be largely measured on how good they are in these areas. Even in the short term pieces of technology like being able to take a voice print of a user and synthesize text in their voice might be applicable. Our parsing work is not too far away from being able to generate a query for "related documents" that can be run on the server or in the backround. Its not clear to me what interaction will be like when all of the new input modalities are highly reliable. Some users might go for using speech for most everything but the days of dictation people would mark changes onto draft documents and point to sections to change.